How to Make your own Neural Network (Part-01)

Dr. Jamshaid Ul Rahman

Prepared by: Malik Ahsin Iqbal & Muhammad Israr

ABDUS SALAM SCHOOL OF MATHEMATICAL SCIENCES
Government College University Lahore

Outline:

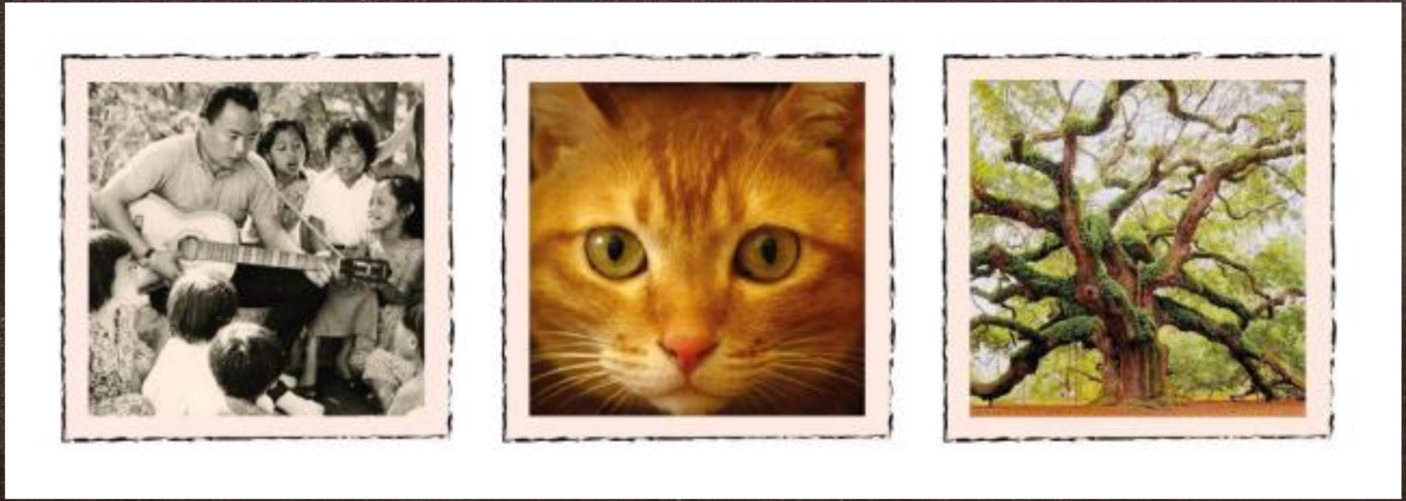
- Comparison Of Human And Computer
- A Simple Predicting Machine
- Classification
- Training a Simpler Classifier
- Learning Rate
- Boolean Logic Function
- Biological Neuron
- Activation Function
- How Neuron Works
- Following Signals Through Neural
- Use Of Matrix Multiplication

Comparison of Computer and Human:

Computers are nothing more than calculators at heart. They are very fast at doing arithmetic.

- Basics Arithmetic operation of thousand or million of number is very easy for computer but it is very difficult for human.

- The recognizing the picture is easy for human but it is difficult for the computer. We can process the quite large amount of information that the images contain, and very successfully process it to recognize what's in the image. For example,

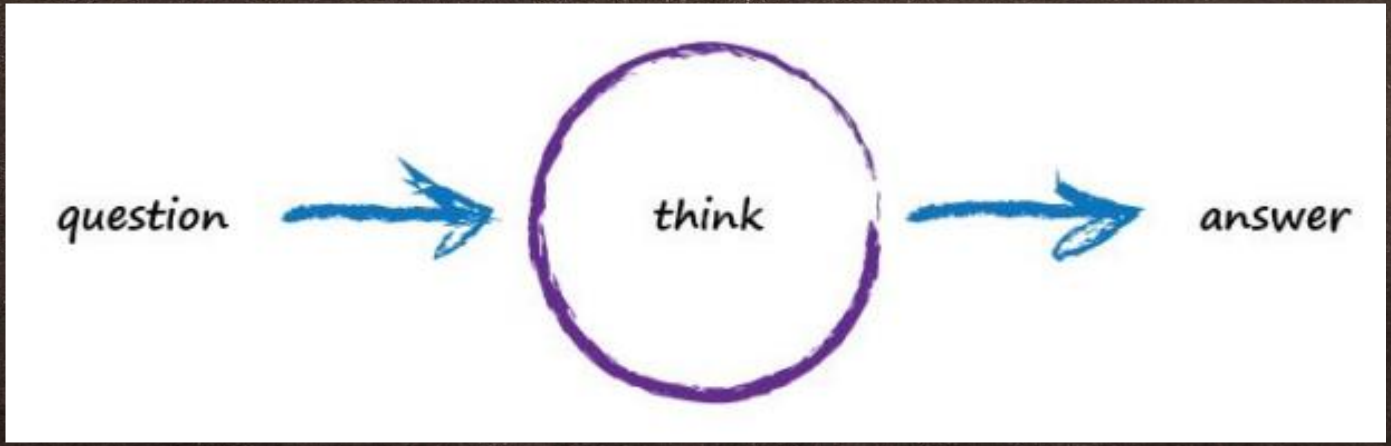


Why we study the Artificial Intelligence ?

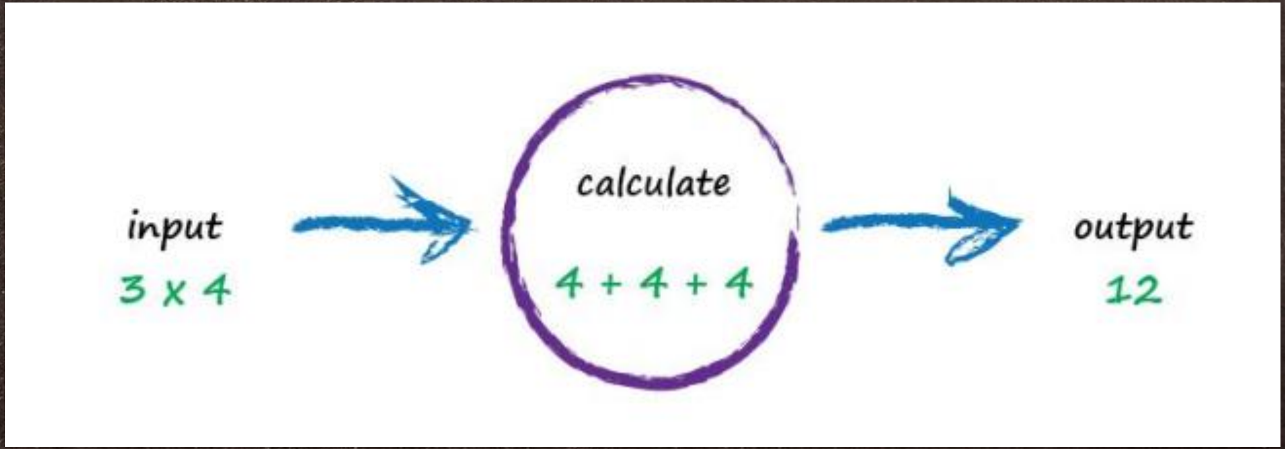
- For these kinds of problems that we want computers to get better at solving because they're fast and don't get tired.
- Computers always made of electronics and so the task of artificial intelligence is to find new kinds of algorithms, which work in new ways to try to solve these kinds of harder problem.

A Simple Predicting Machine:

- Just like the example above with ourselves taking input through our eyes, using our brains to analyze the scene, and coming to the conclusion about what objects are in that scene.

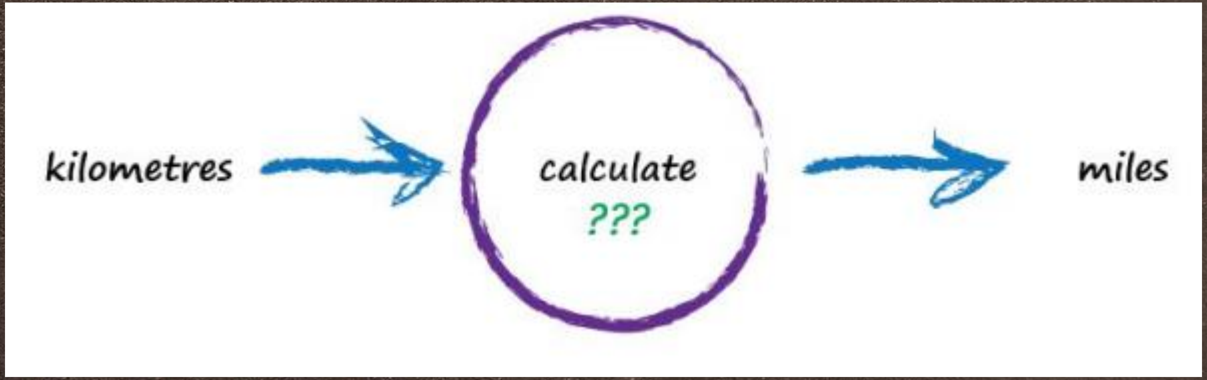


- Imagine a basic machine that takes a question, does some “thinking” and pushes out an answer.
- Computers don’t really think, they’re just glorified calculators remember, so let’s use more appropriate words to describe what’s going on:



Example 2:

Imagine a machine that converts km to miles, like the following:

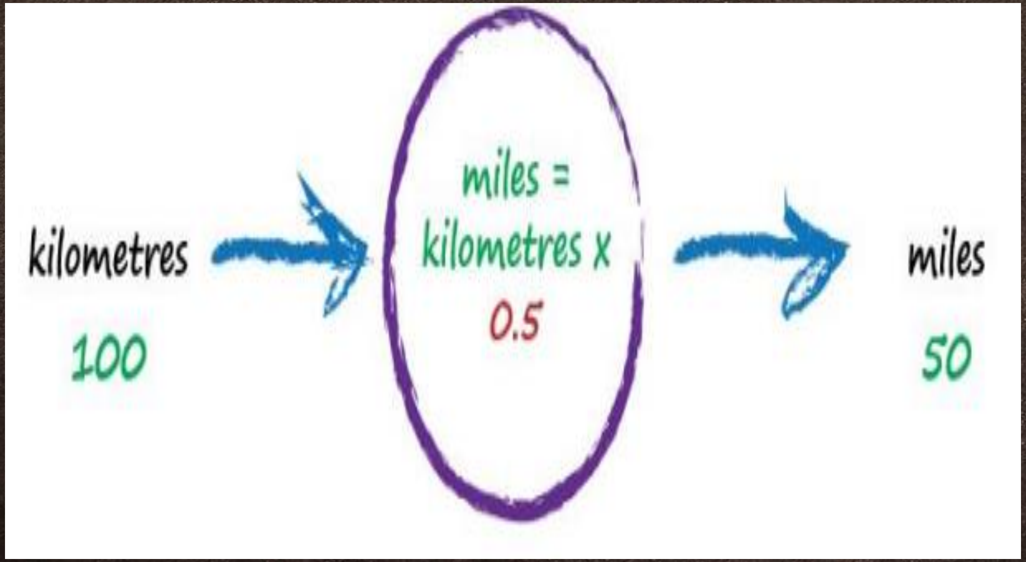


We know that $miles = km * c$, where c is a constant. Let we have the $km = 100$ and $c = 0.5$, than $miles = 50$, but the actual answer is $miles = 62.137$.

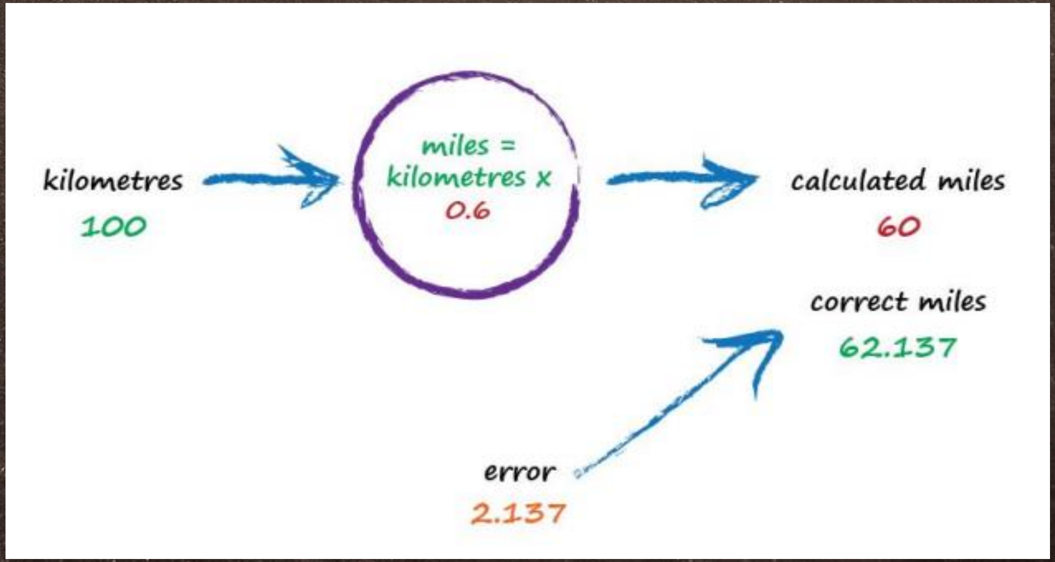
$$Error = 62.137 - 50 = 12.137$$

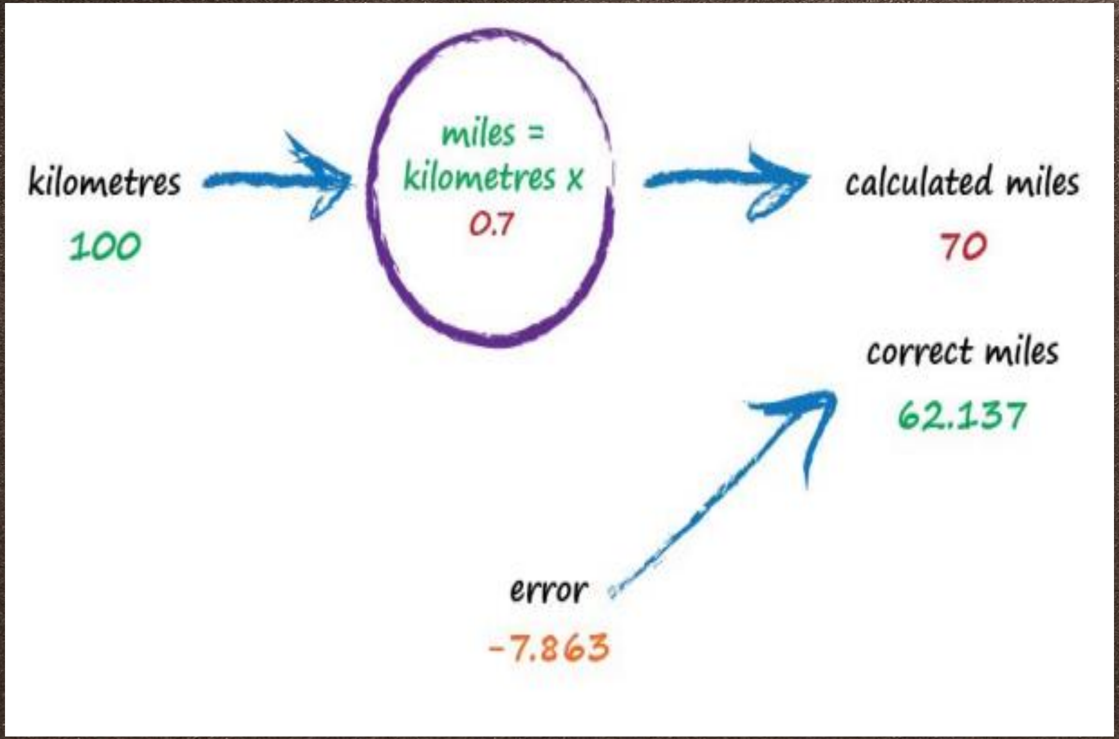
Graphically,

$c = 0.5$



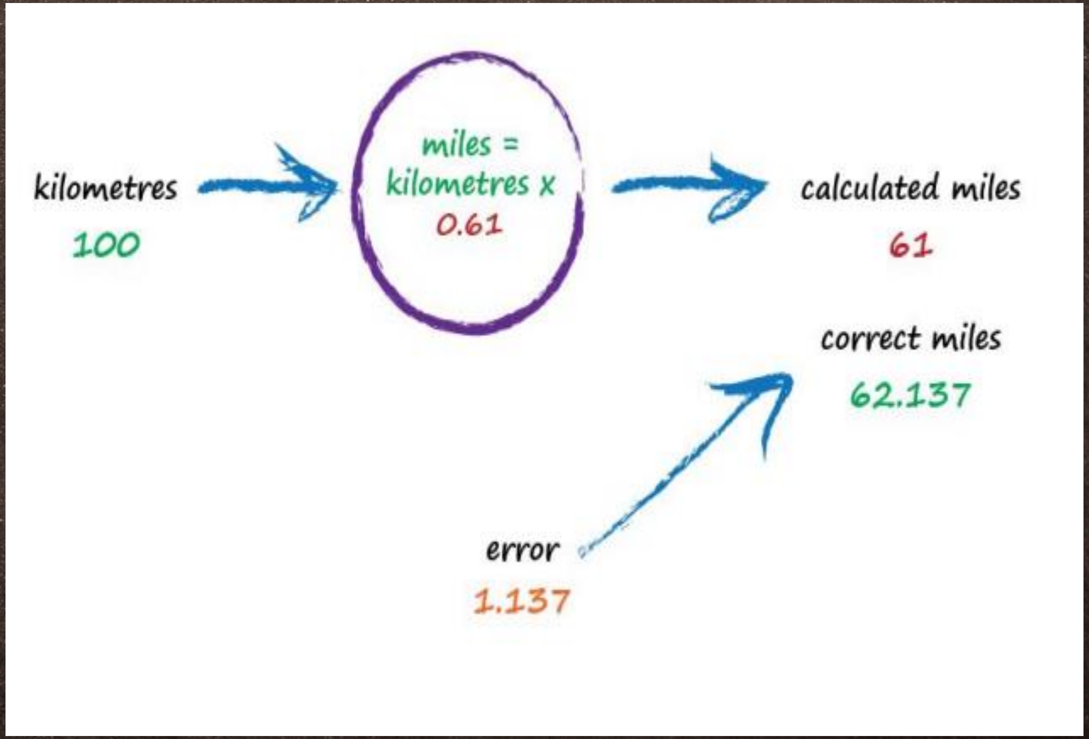
$c = 0.6$




kilometres
100
miles = kilometres x 0.7
calculated miles
70
correct miles
62.137
error
-7.863

$c = 0.7$

$c = 0.61$

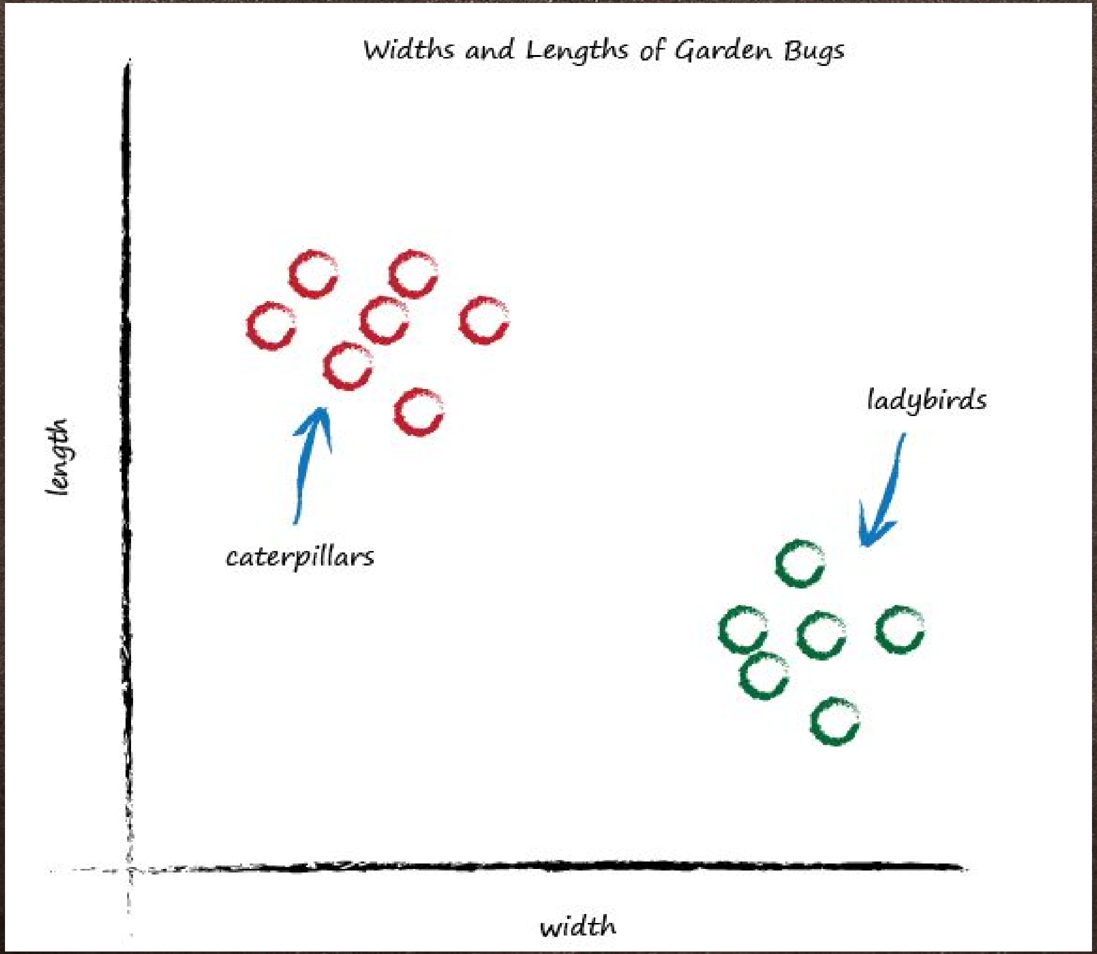

kilometres
100
miles = kilometres x 0.61
calculated miles
61
correct miles
62.137
error
1.137

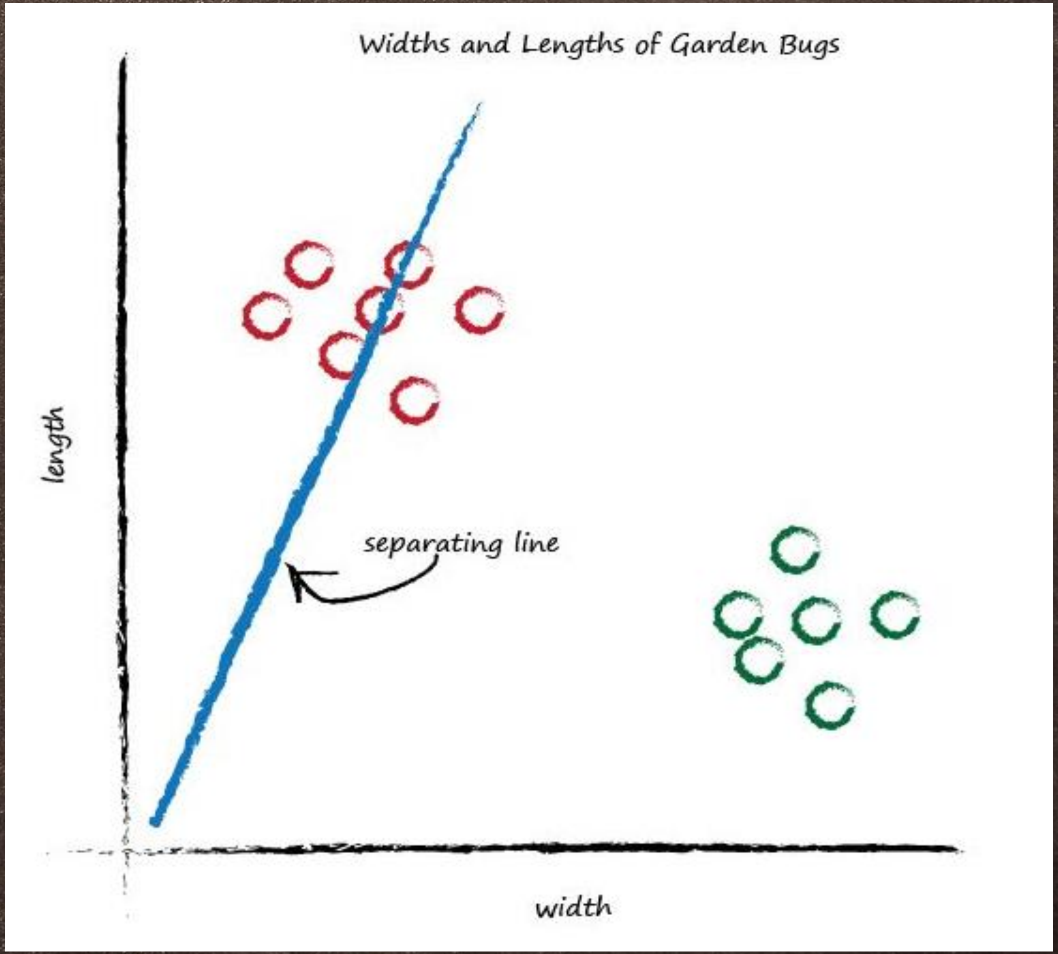
From the above we see that,

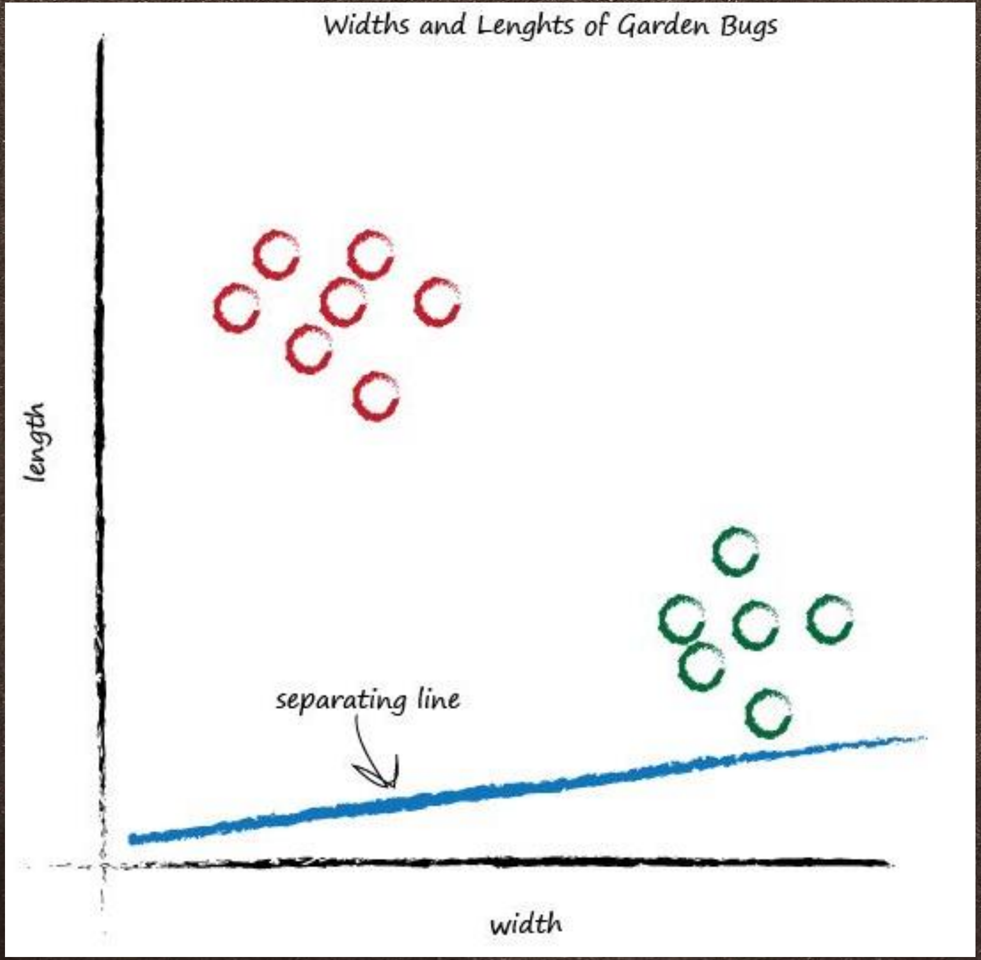
- All useful computer systems have an input, and an output, with some kind of calculation in between. Neural networks are no different.

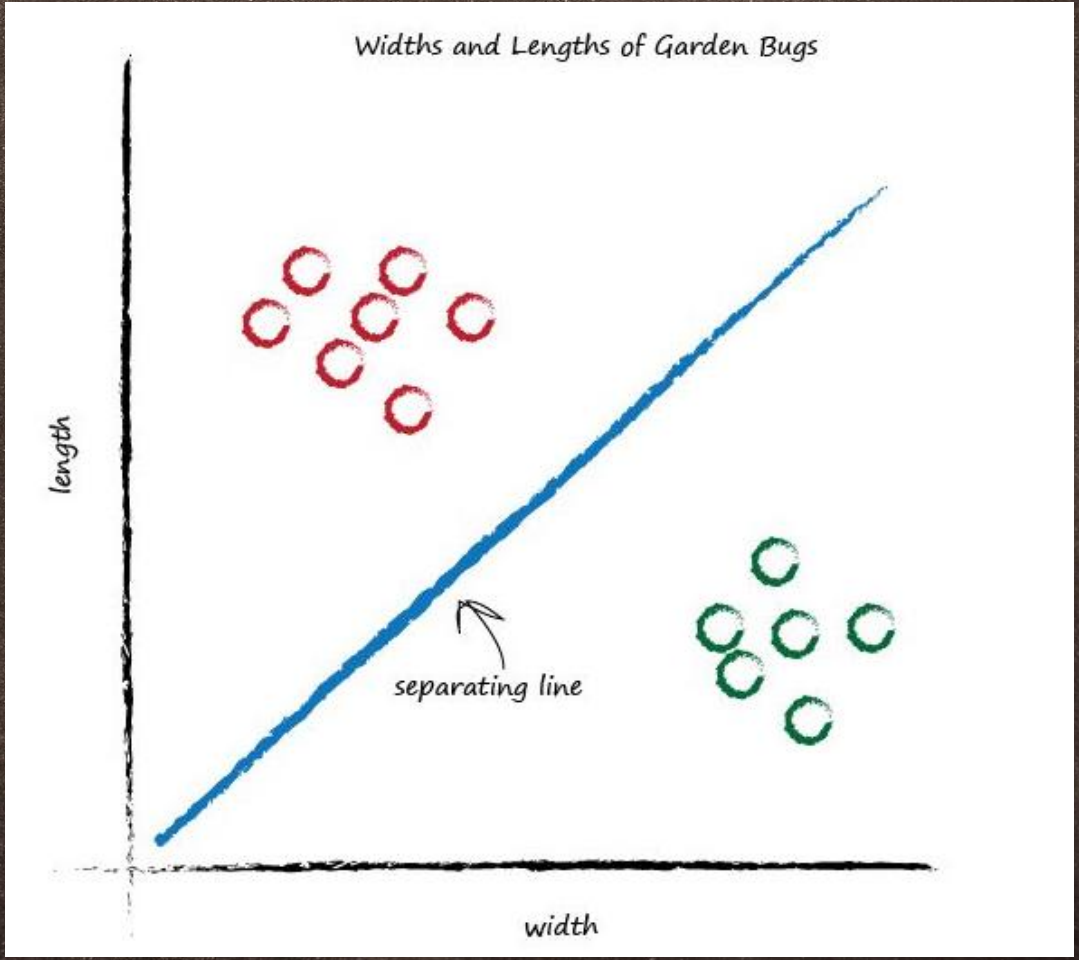
- When we don't know exactly how something works we can try to estimate it with a model which includes parameters which we can adjust. If we didn't know how to convert km to miles, we might use a linear function as a model, with an adjustable gradient.

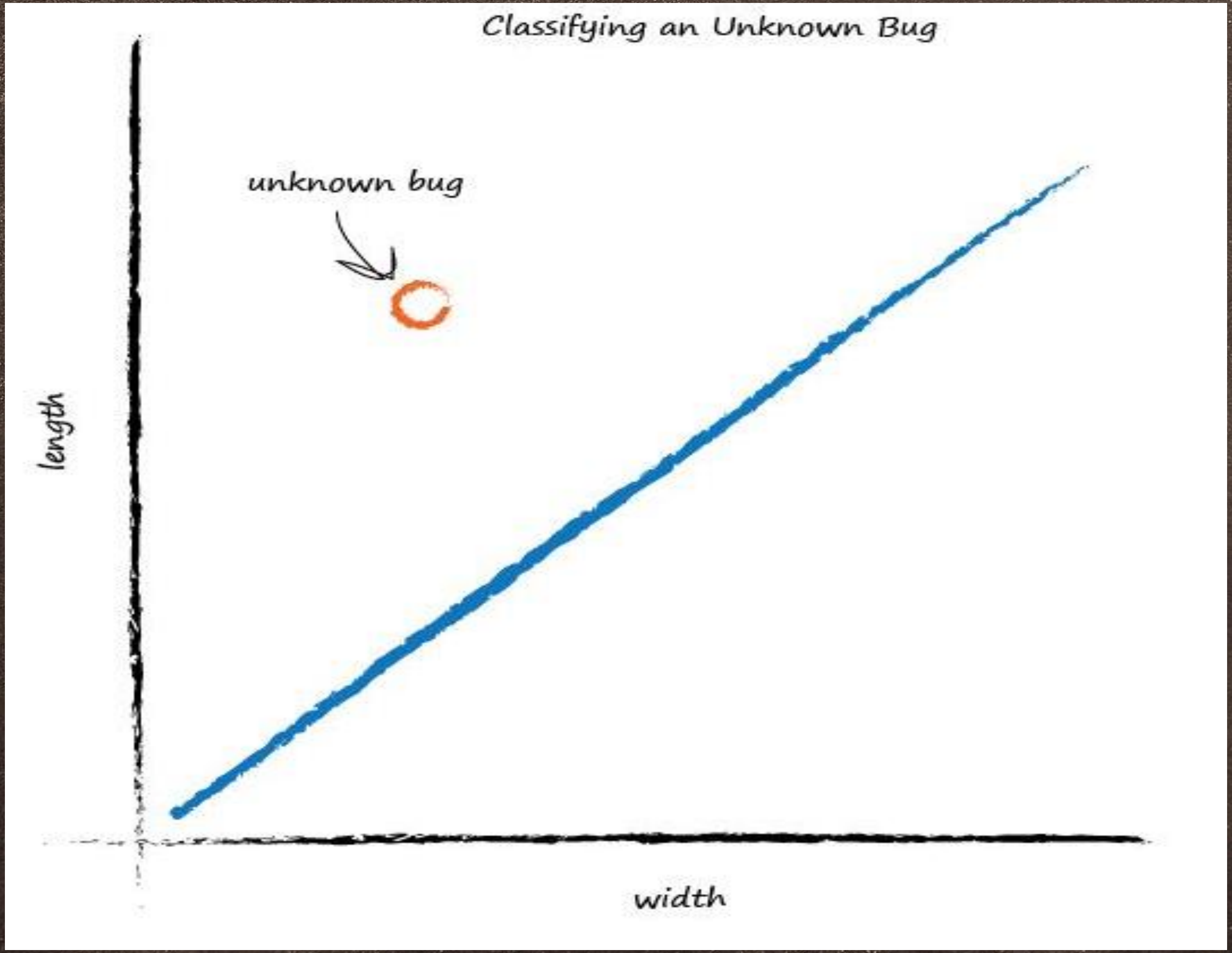
Classifying is not very different from predicting:






Widths and Lenghts of Garden Bugs
length
width
separating line


Widths and Lengths of Garden Bugs
length
width
separating line

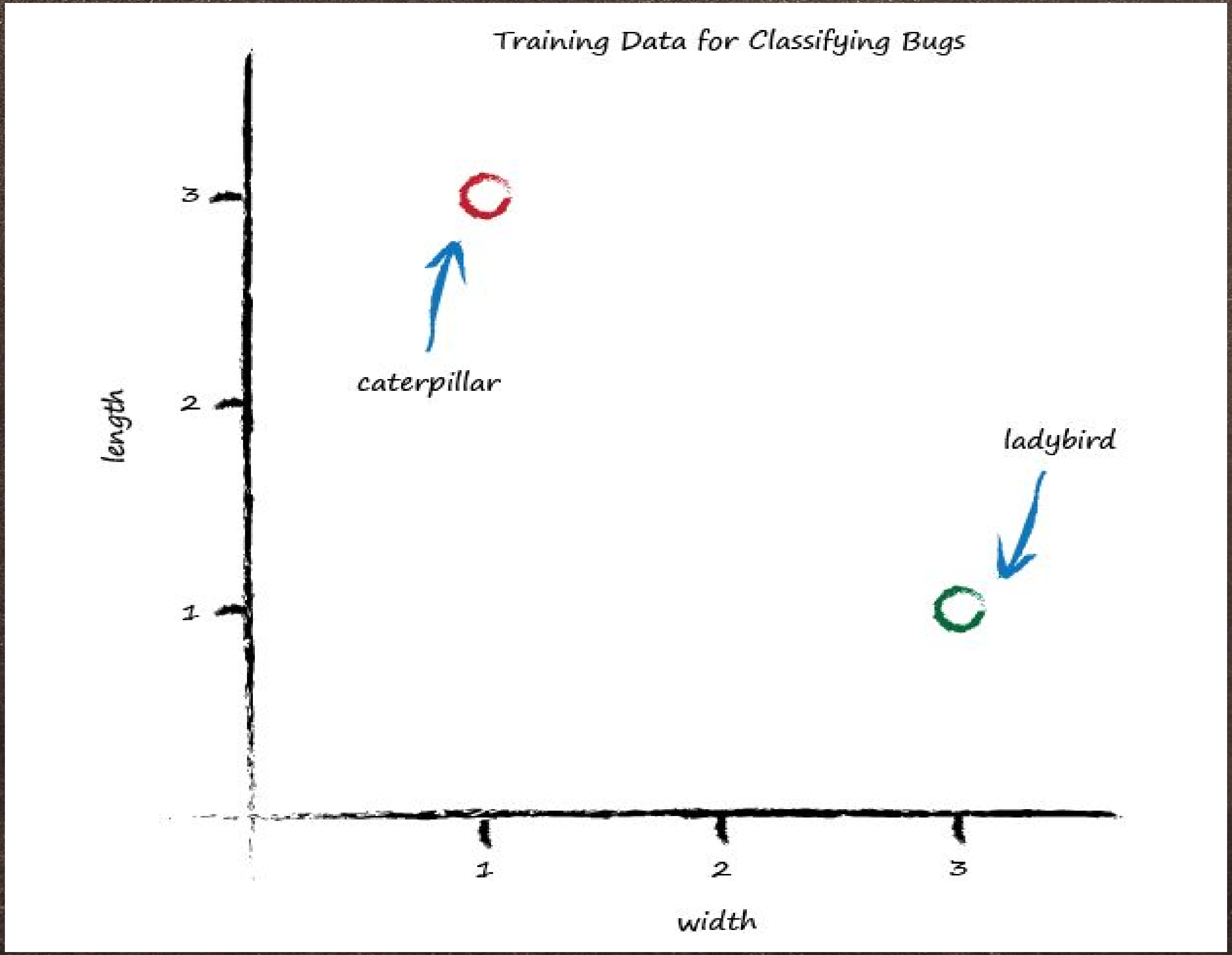

Classifying an Unknown Bug
unknown bug
length
width

Training a simpler classifier:

From above we can see that,
To train our classifier we can change the slope of line that divides the two groups.
Consider the example,

Example	Width	Length	Group
1	3.0	1	Ladybird
2	1.0	3.0	caterpillar

Note: Example of truth used to teach the predictor or classifier are called the training data.

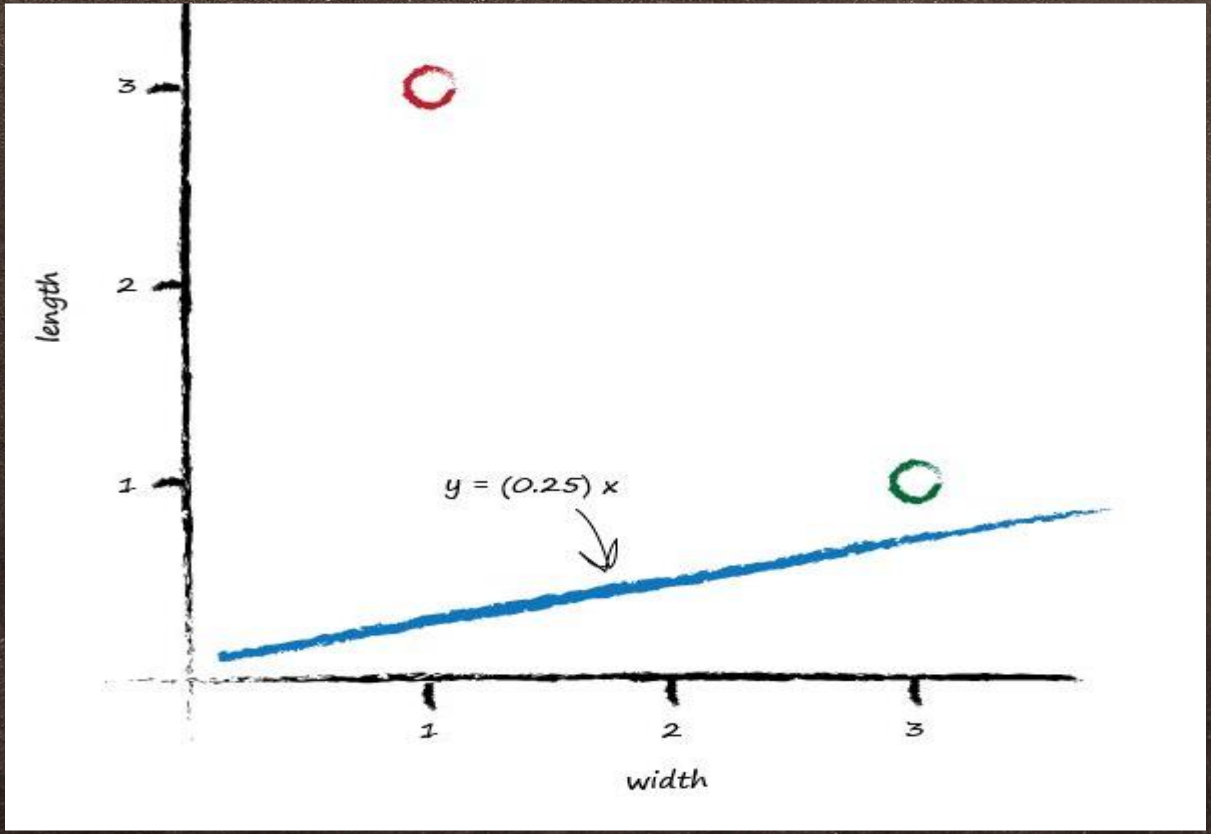


Now consider the equation of line,

$$y = Ax + c, where\ A\ \&\ c\ are\ consatnt$$

For our purpose we take $c = 0$.

Let suppose $A = 0.25$, then $y = 0.25x$.



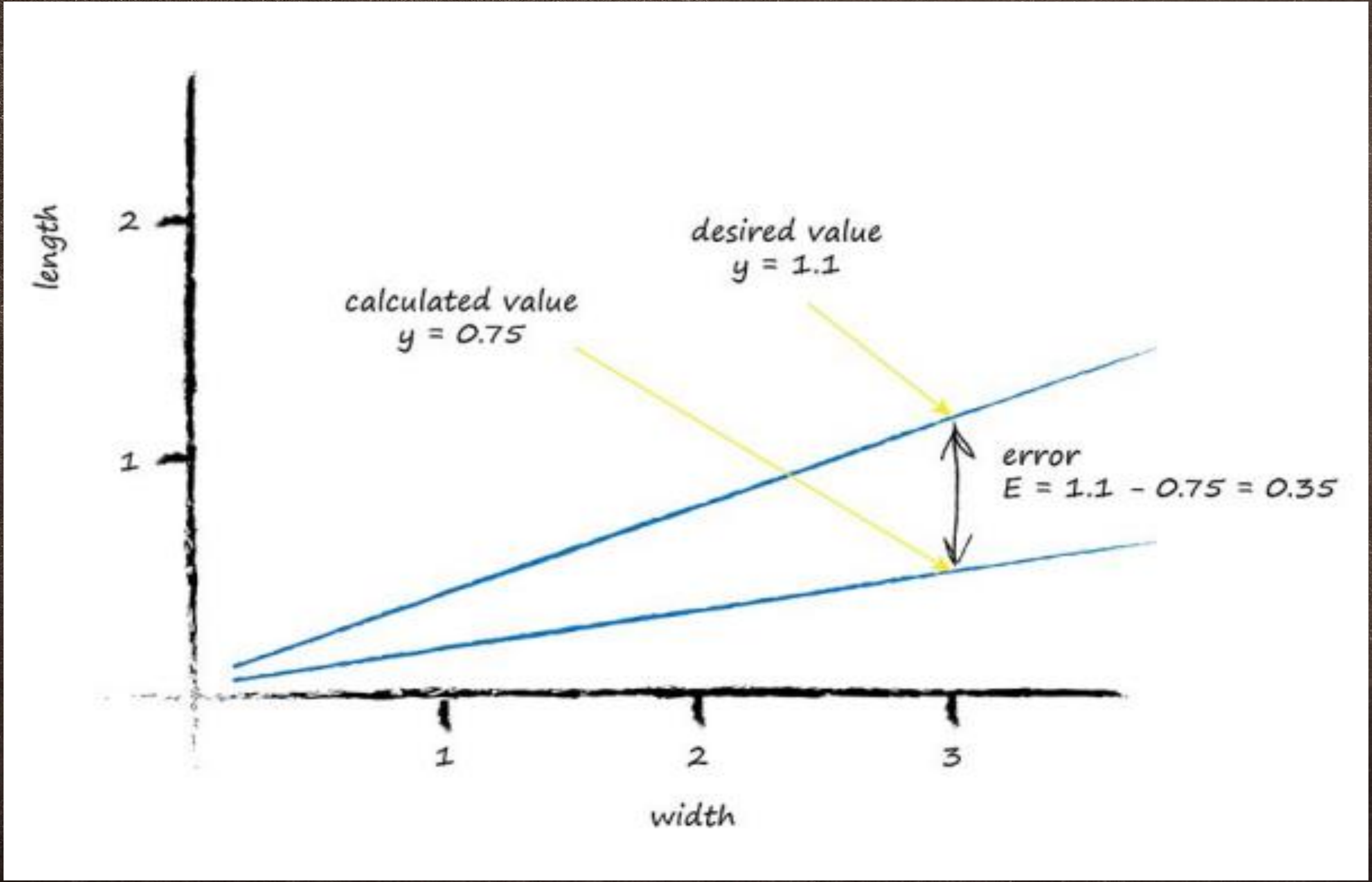
Look at the 1st example,
the width is 3.0 and length is 1 for a ladybird. If we tested the $y = Ax$ function with this example where x is 3.0, we get

$$y = 0.25 * 3.0 = 0.75$$

Here we choose A randomly, but we know that the actual value at $x = 3$ is 1.
And we have an error i.e.

$$E = 1.1 - 0.75 = 0.35$$

Graphically,

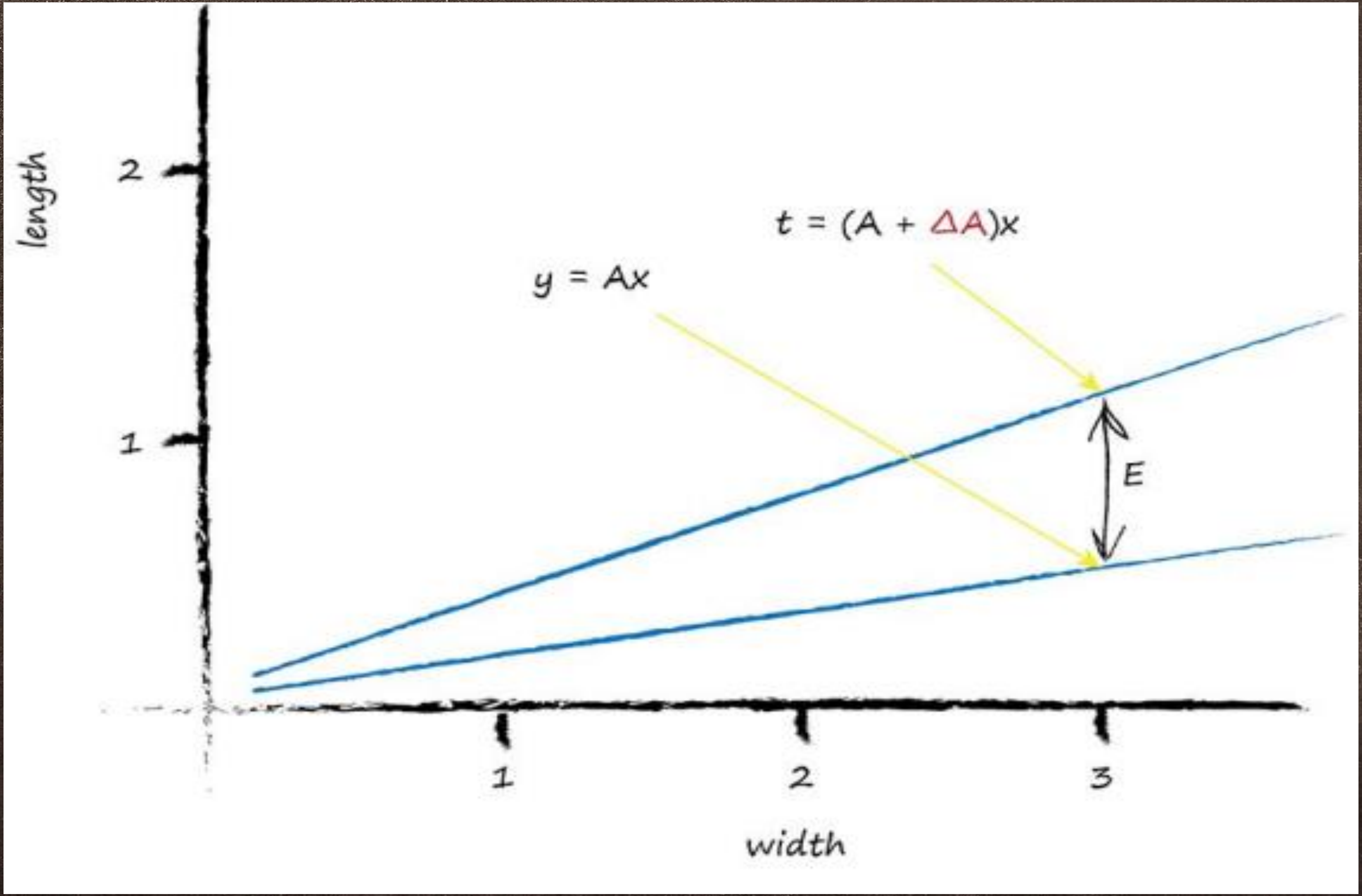


We want to use the error in y, and to check the required change in A.
Our goals is to know how A is related to E ?
If we can know this, then we can understand how change in one affects the other.
Now we have $y = Ax$, for $A = 0.25$ we get the wrong answer, to make a small change in A, i.e.

$$t = (A + \Delta A)x$$

Here t is used for targeted or actual value.

Let see this with the help of graph.



we know that $E = actual - approximate$

$$E = t - y = (A + \Delta A)x - Ax = \Delta Ax$$

So, $$\Delta A = \frac{E}{x}$$

As $E = 0.35, x = 3$ then $\Delta A = 0.1167$.

We need to change the current $A = 0.25$ by $\Delta A = 0.1167$, that is, $A + \Delta A = 0.3667$

And we get $y = 0.3667 * 3 = 1.1$.

Similarly if we consider the 2nd example where $x = 1.0$ and we take updated value of A, then

$$y = 0.3667 * 1 = 0.3667$$

We have the desired targeted value is $\mathrm{t} = 2.9$, so

$$E = 2.9 - 0.3667 = 2.5333$$

Updating in similar way,

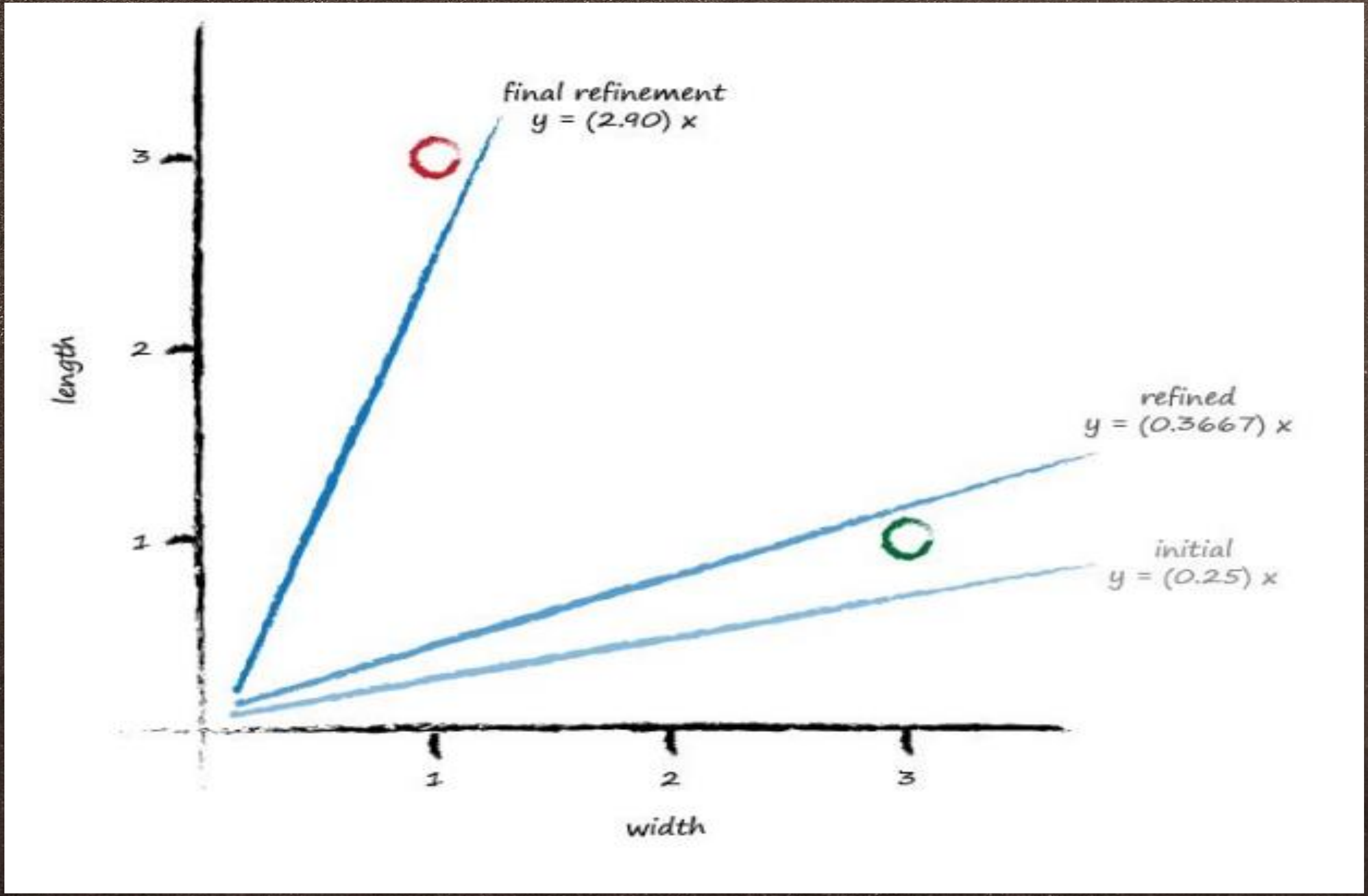
$$\Delta A = \frac{E}{x} = \frac{2.5333}{1} = 2.5333$$

So the new value of

$$A = 2.5333 + 0.3667 = 2.9$$

Hence for $x = 1, y = 2.9$

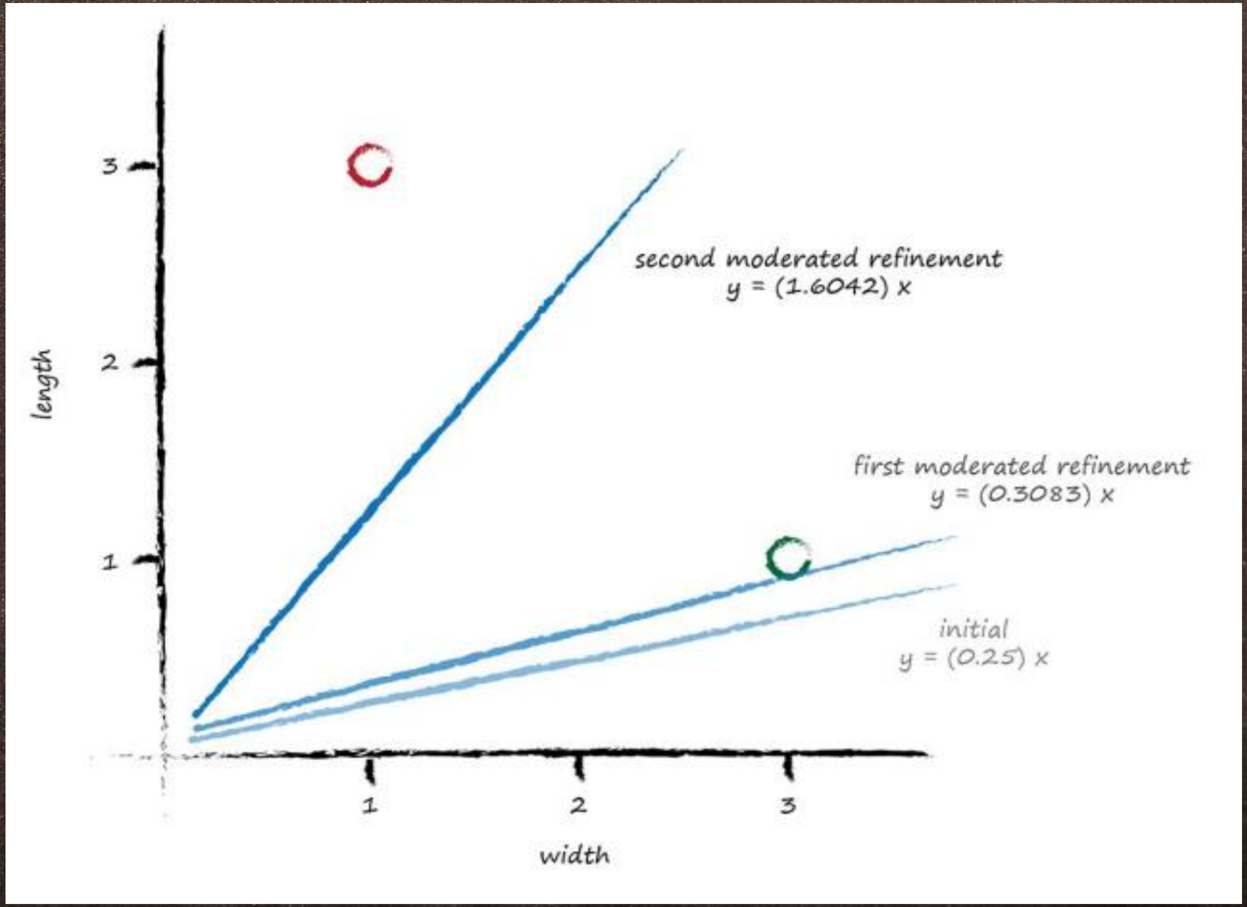
Graphically,



Learning Rate:

The moderating factor $\Delta A = \frac{E}{x}$ is called the learning rate.

Graphically,

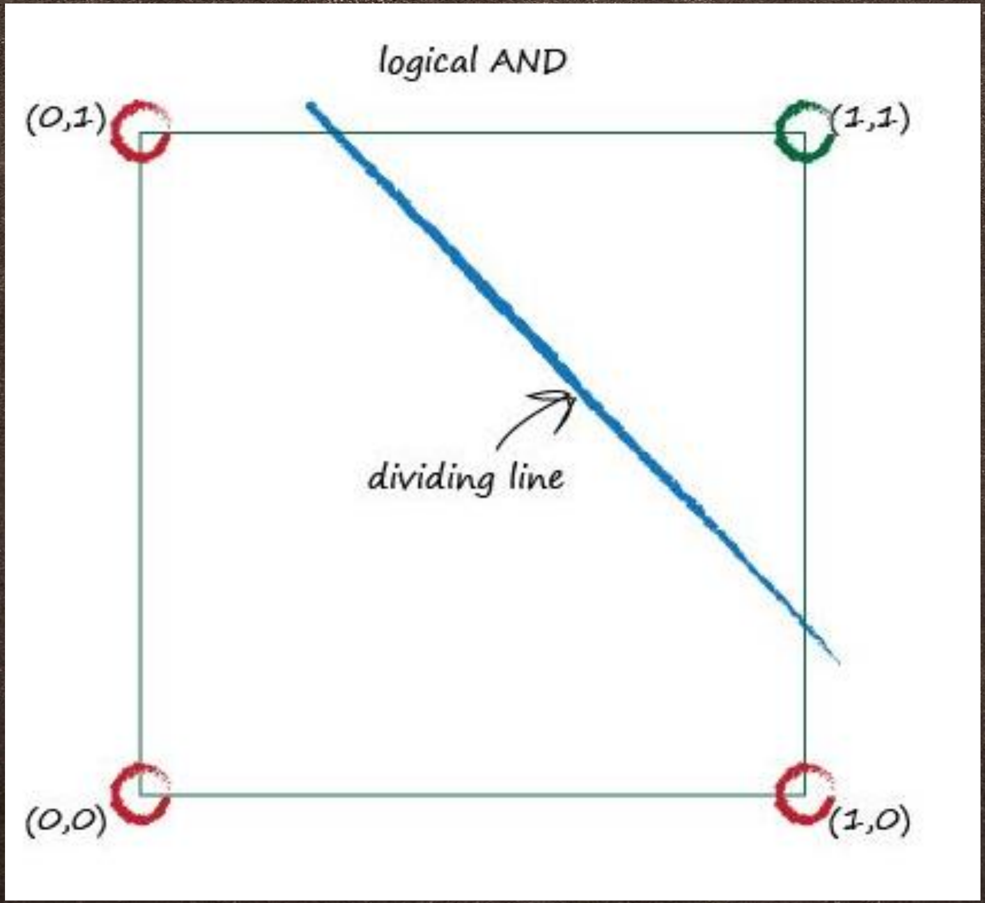


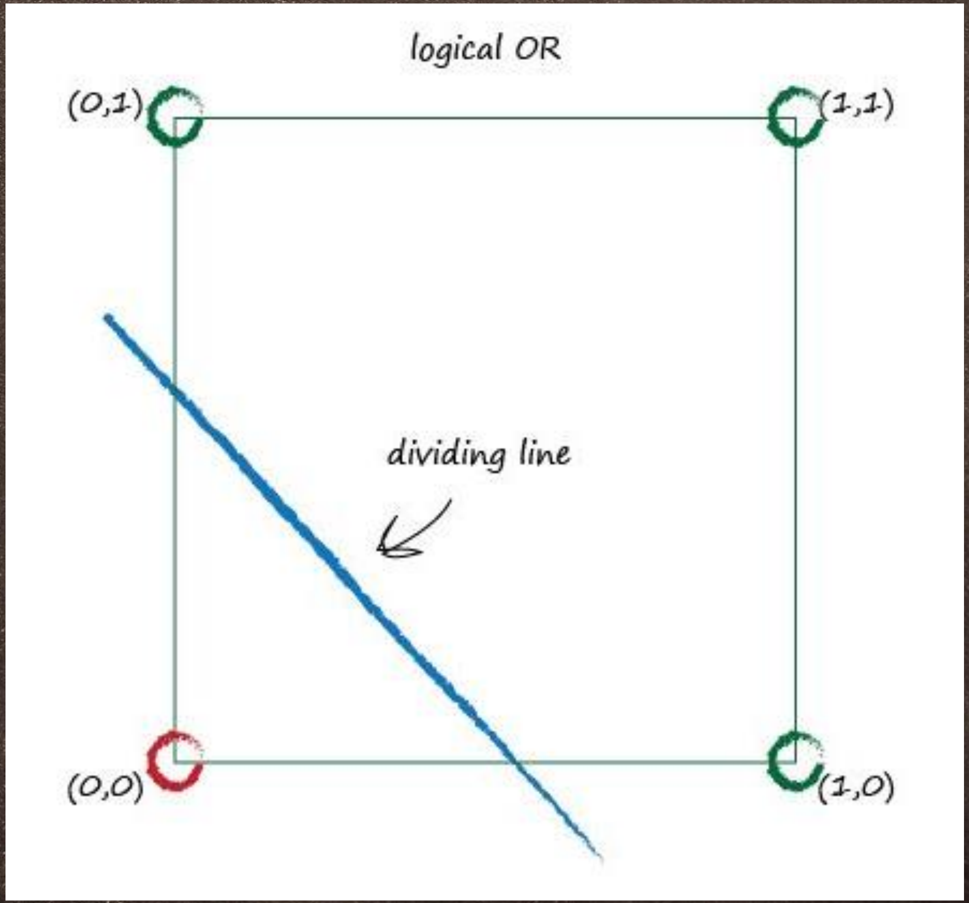
Boolean Logic Function:

Here we used 1 for true and 0 for false. We understand these function with the help of the table, here we use 1 for True and 0 for False

Input A	Input B	Logical AND	Logical OR
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

Graphs of Boolean functions:





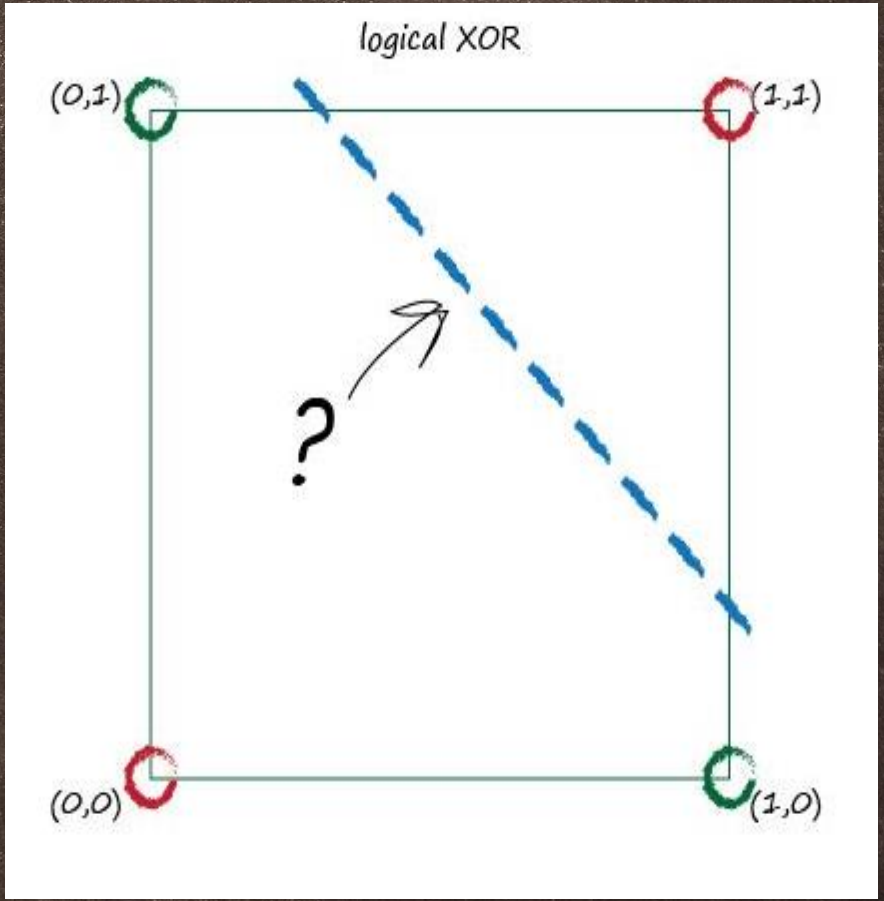
The beauty of above graph is that it make clear that it is possible for linear classifier to learn the Boolean Functions AND & OR.

The Logical Function $\mathbf{XOR}$:

XOR is short form of the eXclusive OR. Which only has a true output if either one of the inputs A or B is true, but not both. That is, when the inputs are both false, or both true, the output is false.

Table of Logical operator XOR, and Classifier

Input A	Input B	Logical XOR
1	1	0
1	0	1
0	1	1
0	0	0



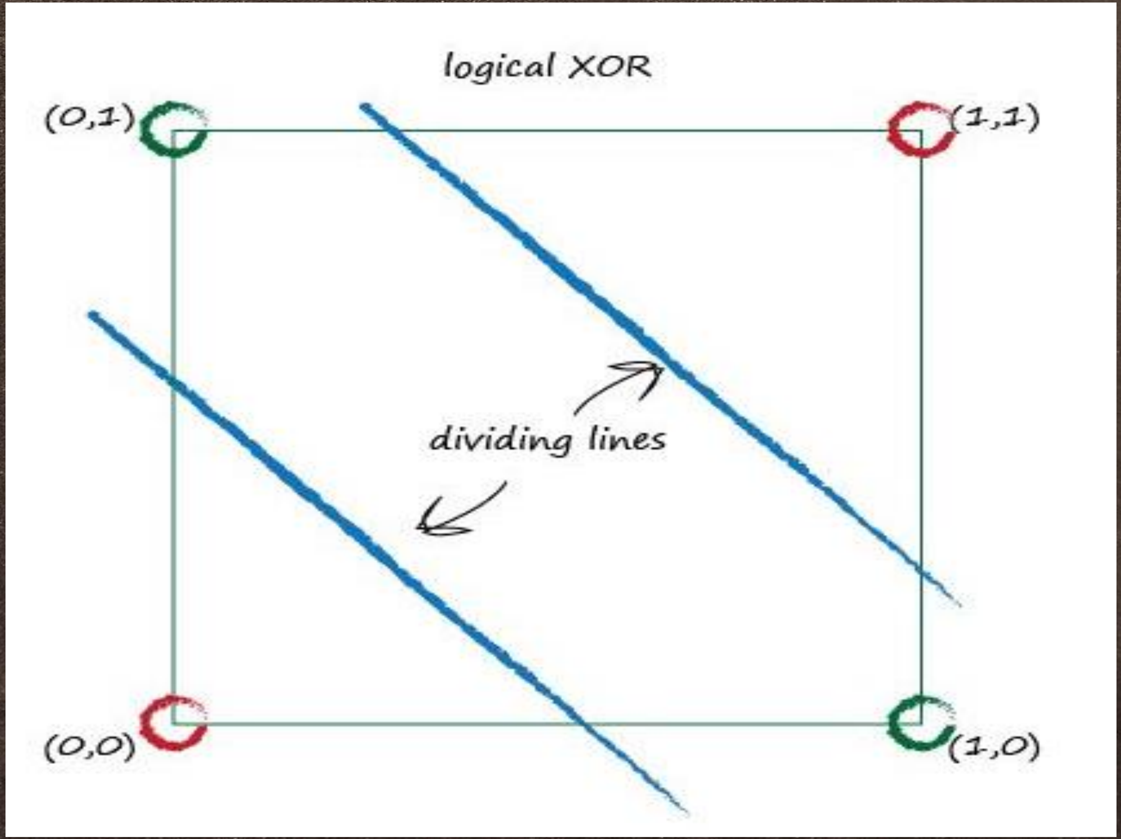
Limitation of Linear classifier:

- It is, in fact, impossible to have a single straight line that successfully divides the red from the green regions for the Boolean XOR. That is, a simple linear classifier can't learn the Boolean XOR if presented with training data that was governed by the XOR function.
- A simple linear classifier is not useful if the underlying problem is not separable by a straight line.

Solution for above problem:

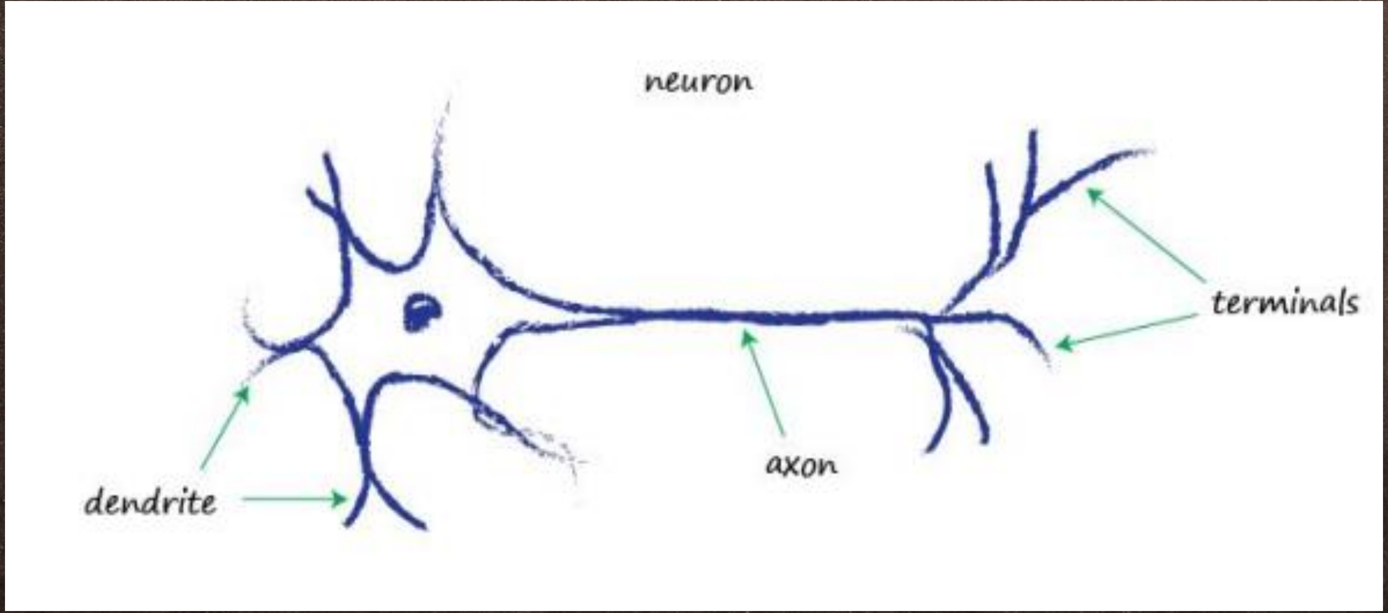
To over come this problem we use multiple linear classifier to separate the colors.

Graphically,

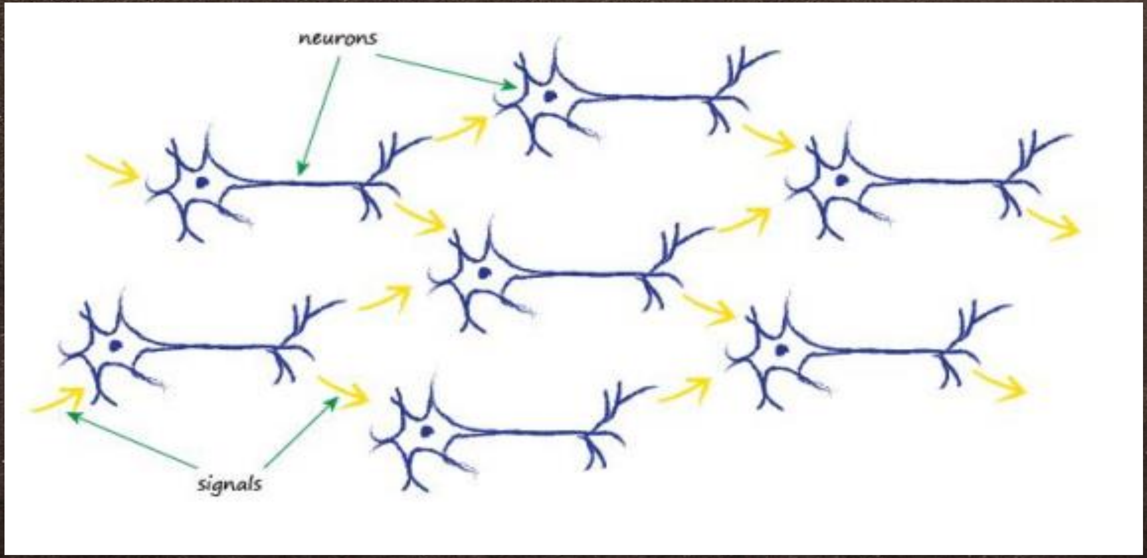


Biological Neuron:

All neuron transmit electrical signals from one end to another end, from the dendrites through axon to terminal. These signals are then passed from one neuron to another neuron.



The electrical signals are collected by the dendrites and these combine to form a stronger electrical signal. If the signal is strong enough to pass the threshold, the neuron fires a signal down the axon towards the terminals to pass onto the next neuron's dendrites.



How neuron works:

Neuron takes an electrical signal and give another electrical signal. Just like the simple predicting machine.

Threshold value:

The threshold value is a parameter in the neuron model, allowing for adjustments during the training process to better fit the neuron's behavior to the desired output.

Mathematically,

if $x_1, x_{2,} \ldots x_n$ are inputs to the neuron and $w_1, w_2, \ldots, w_n$ are the corresponding weights, and b is the threshold value than,

$$y = \begin{cases} 1, & if \sum_{i=1}^{n} (w_i x_i) \geq b \\ 0, & otherwise \end{cases}$$

Activation Function:

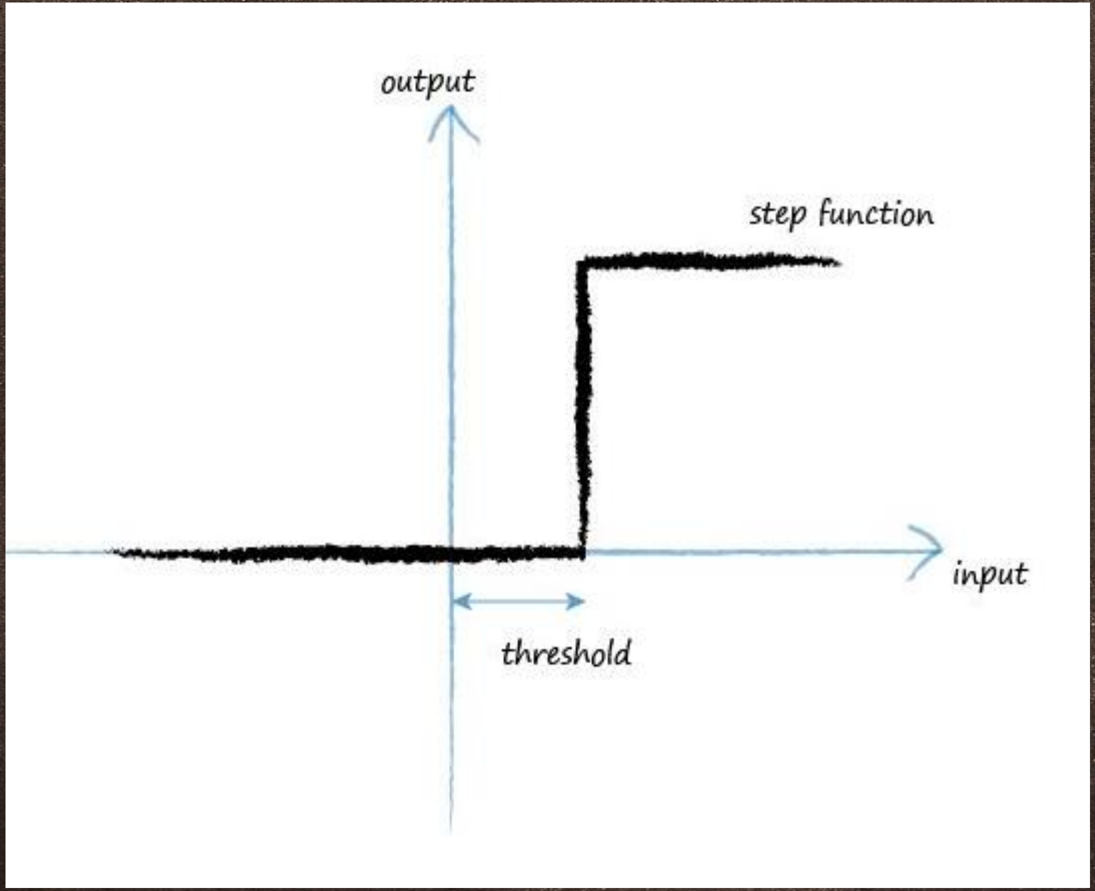
A function that takes the input signal and generates an output signal, but takes into account some kind of threshold is called an activation function. Some of them are,

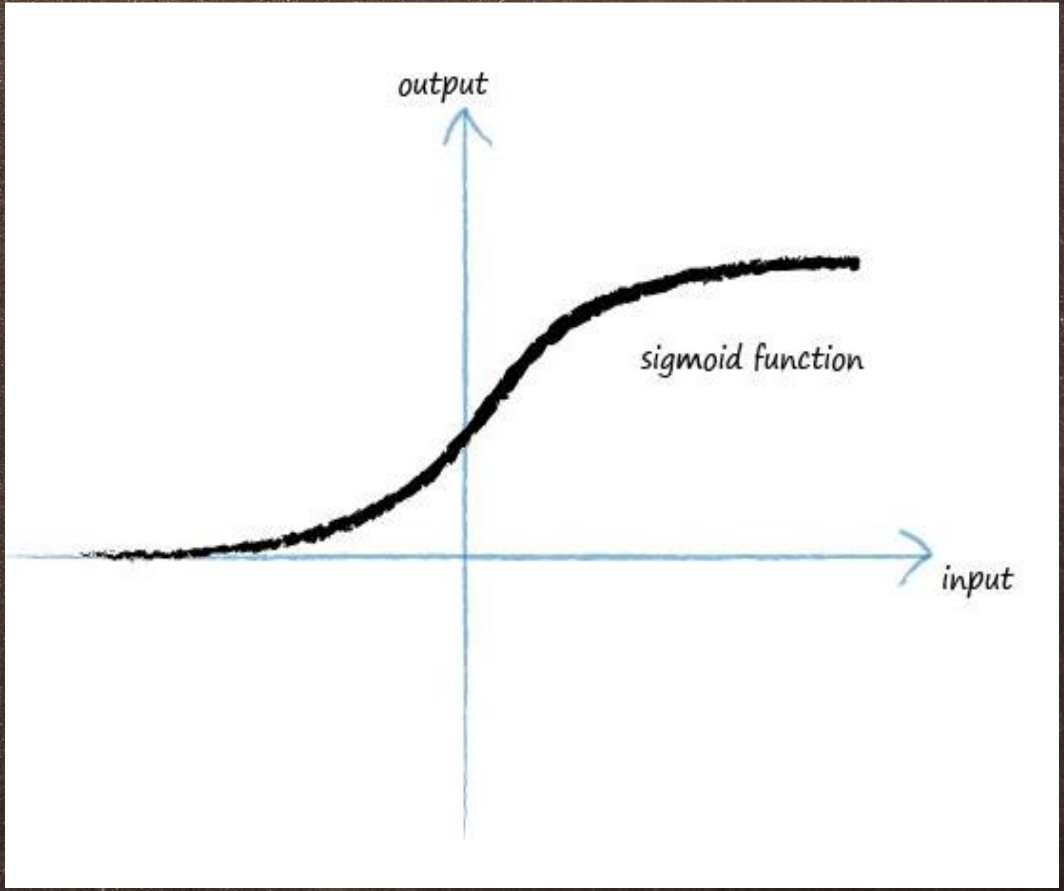
Step and Sigmoid Activation Function:

Let $net = \sum_{i=1}^{n}\left(w_{ij}x_i\right) + b$,

$$y = \begin{cases} 1, & if\ net \geq 0 \\ 0, & if\ net < 0 \end{cases} \quad \& \quad y = \frac{1}{1+e^{-x}}$$

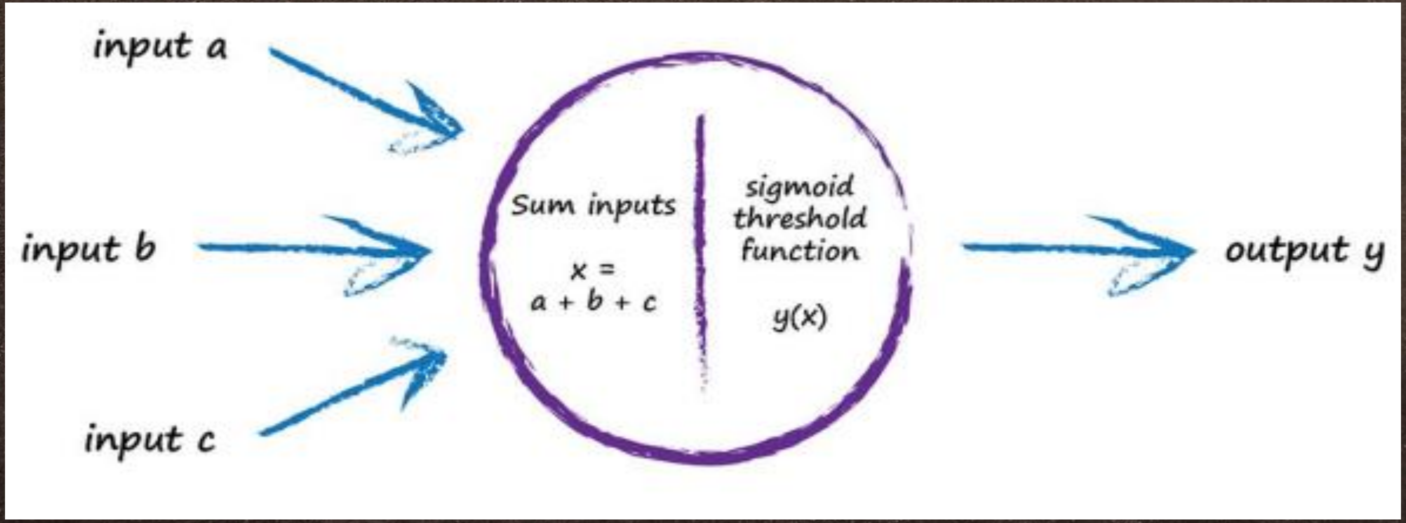
Step function and Sigmoid Function:





How neuron works:

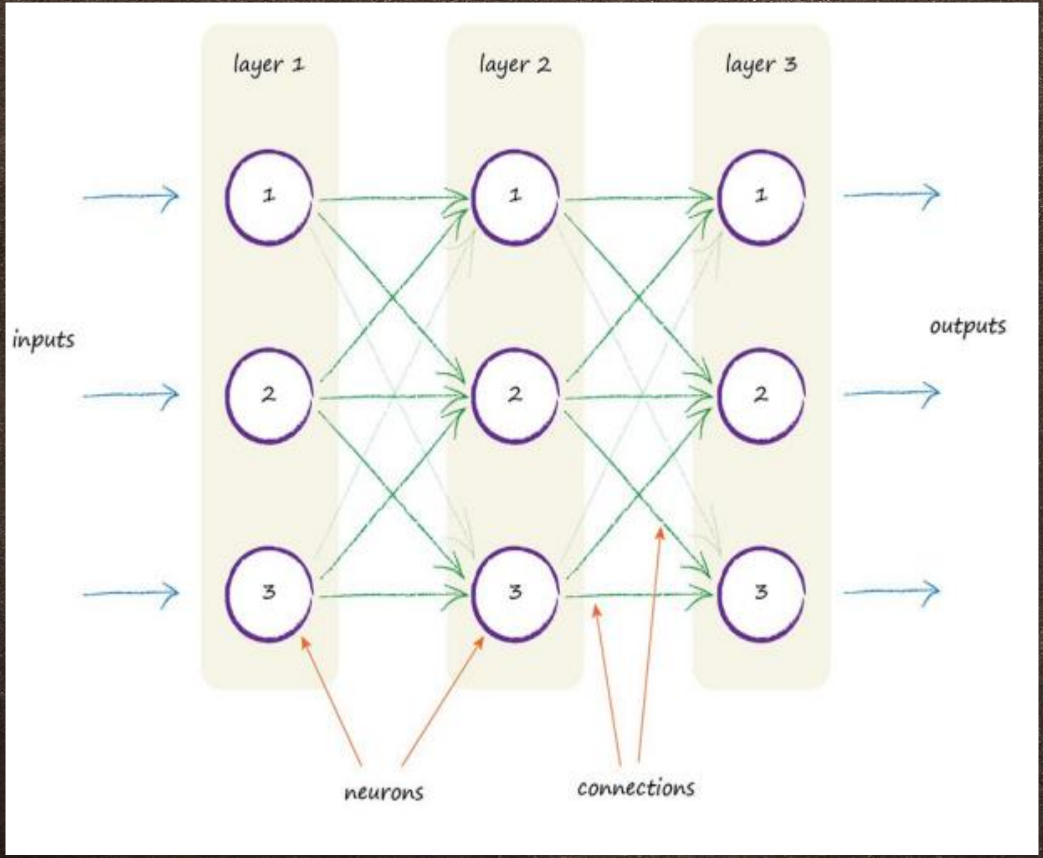
We see in Boolean logic machine that we may have more than one inputs. We simply combine them by adding them up, and the resultant sum is the input to the sigmoid function which controls the output.

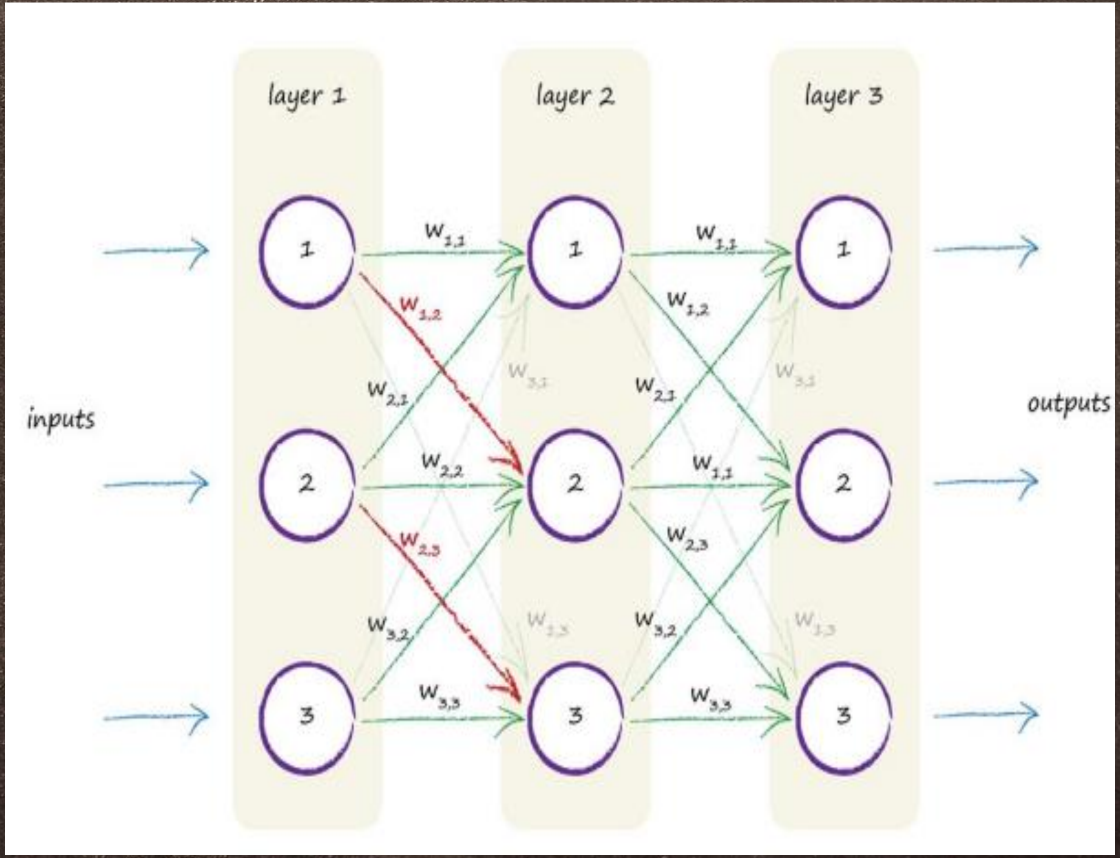


Some Key points about Neuron:

- If the combined signal is not large enough then the effect of the sigmoid threshold function is to suppress the output signal.
- If the sum x is large enough the effect of the sigmoid is to fire the neuron.
- if only one of the several inputs is large and the rest small, this may be enough to fire the neuron

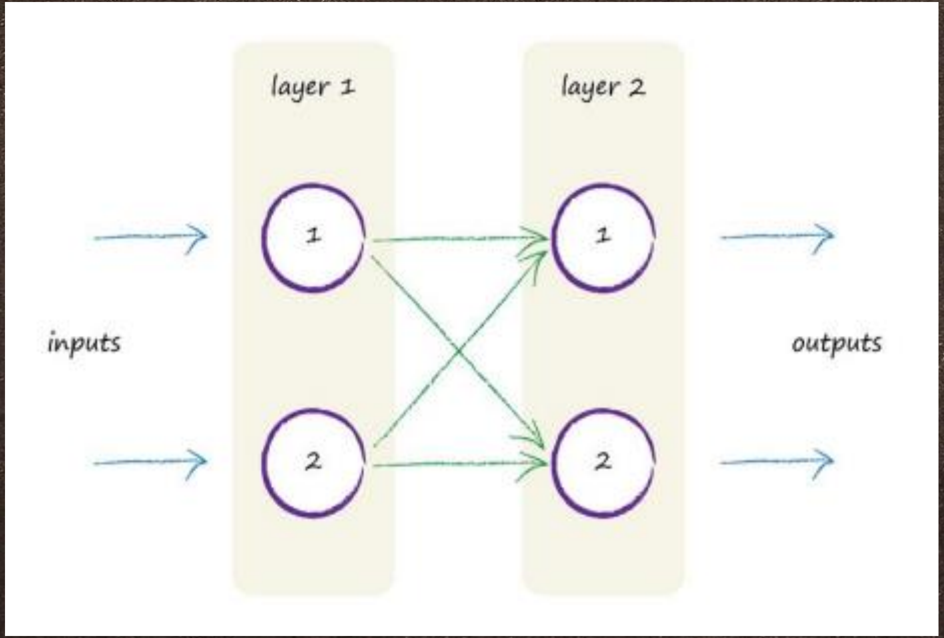
Layers of neurons:

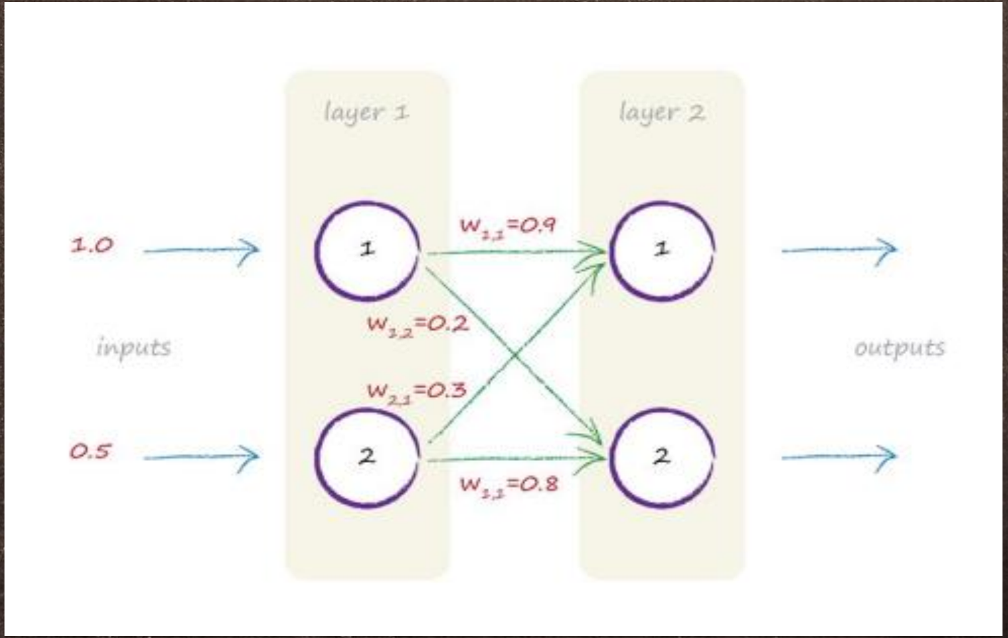




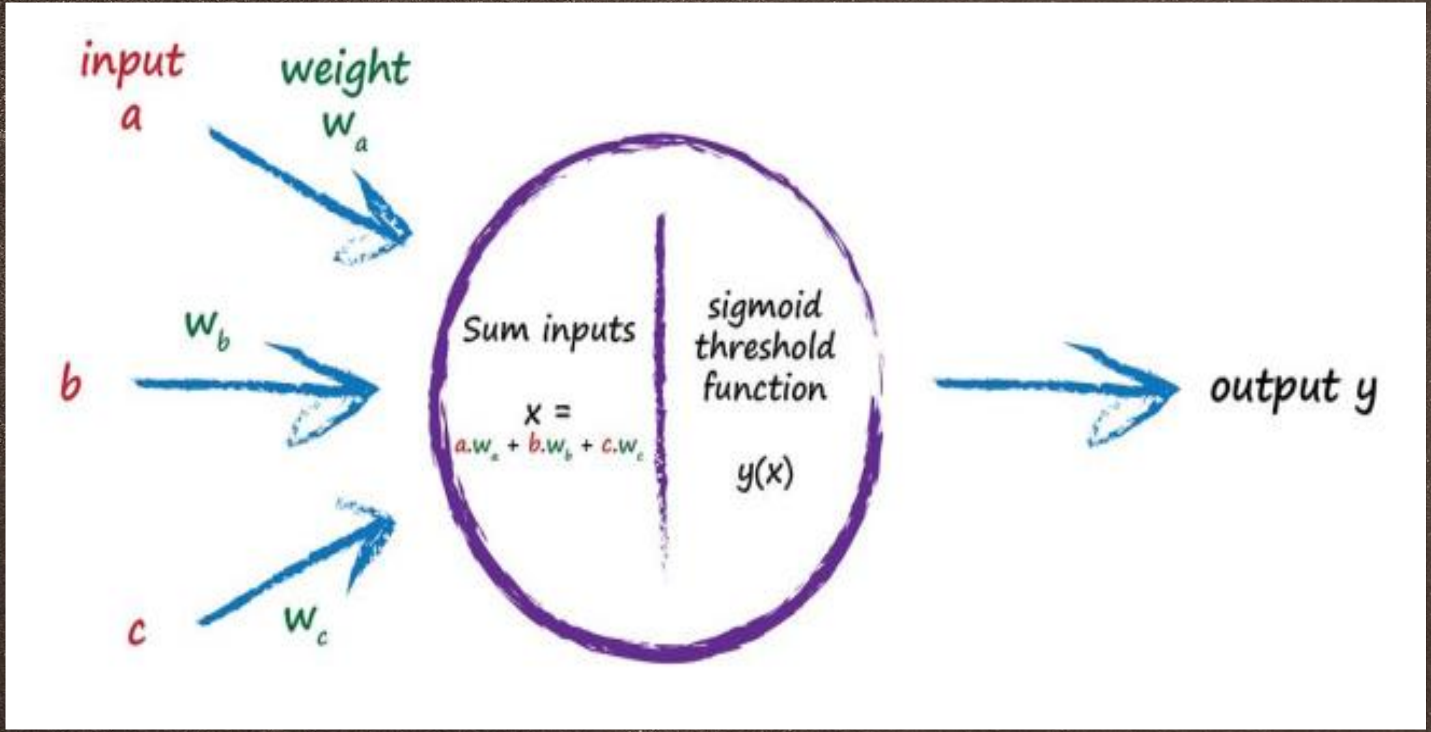
Following Signals Through A Neural Network:

Our next goals is to study how signals progress from the inputs through the layers to become the outputs. For this consider the neural network with 2 layers i.e.





Now see this diagram,



For previous example,

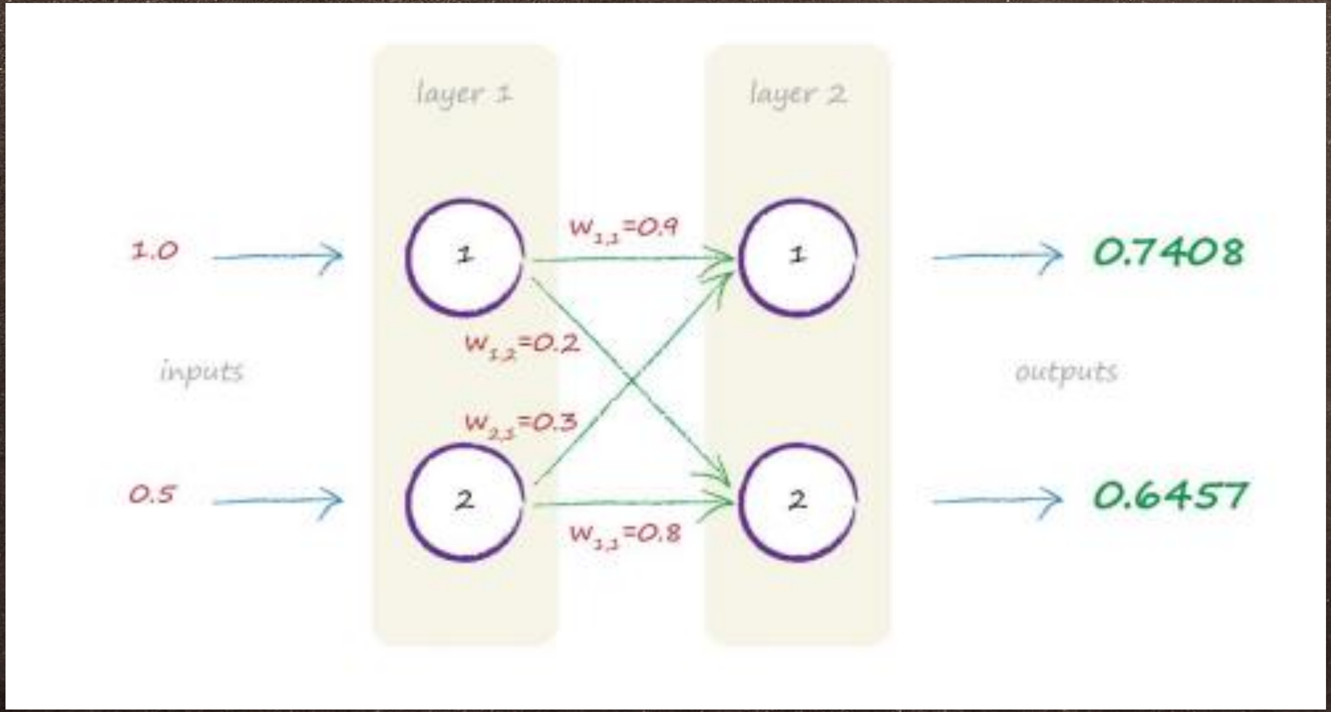
$$x = (1 * 0.9) + (0.5 * 0.3) = 1.05$$

As $y = \frac{1}{1+e^{-x}}$

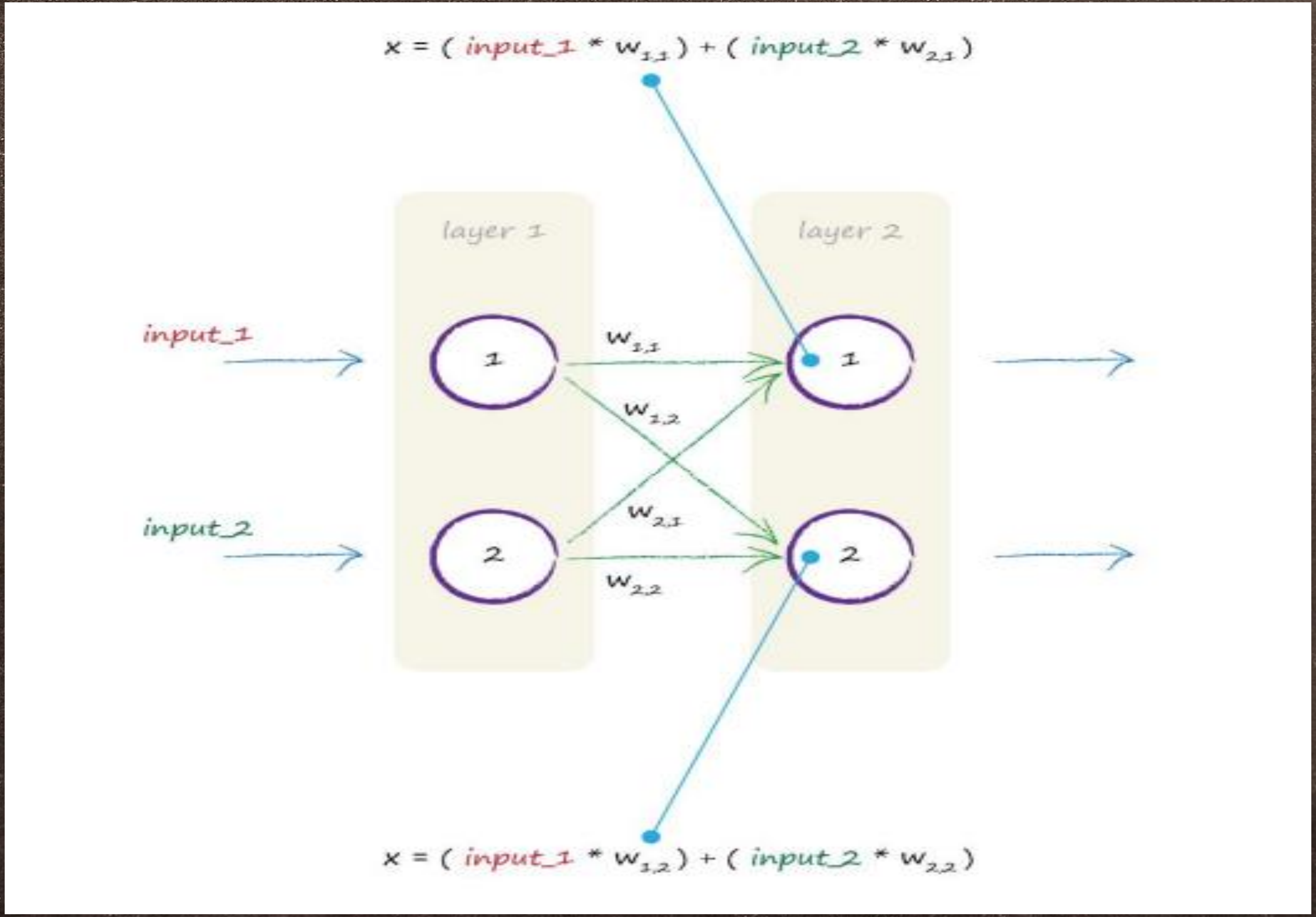
Putting $x = 1.05,\ \ y = 0.7408$

Similarly, for $x = (1.0 * 0.2) + (0.5 * 0.8) = 0.6$

$$y = 0.6457$$



Use of matrix multiplication:


$x = (input\_1 * w_{1,1}) + (input\_2 * w_{2,1})$
layer 1
layer 2
input_1
input_2
1
2
1
2
$w_{1,1}$
$w_{1,2}$
$w_{2,1}$
$w_{2,2}$
$x = (input\_1 * w_{1,2}) + (input\_2 * w_{2,2})$

In matrix form we can write,

$$\begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} w_{1,1} * a + w_{2,1} * b \\ w_{1,2} * a + w_{2,2} * b \end{pmatrix}$$

Where a=input_1 and b=input_2.

The first matrix contains the weights between nodes of two layers. The second matrix contains the signals of the first input layer. The answer we get by multiplying these two matrices is the combined moderated signal into the nodes of the second layer.

In general we can write,

$$X = W.I$$

Where X is resultant matrix, W is weights matrix & I is input matrix.

And output can be written as,

$$O = sigmoid(X)$$

Where O is the output matrix.

Part I end

Acknowledgment:

This presentation was prepared by Malik Ahsin Iqbal and Muhammad Israr Mphil Scholars at ASSMS Batch 22, during the course “Foundations in Deep Learning ” taught by Dr. Jamshaid UL Rahman. These slides cover half of the theoretical part of the book titled “Making your Own Neural Network” by Tariq Rashid.